APPENDIX F
TRIM.FaTE Computer Framework

The TRIM.FaTE computer framework provides the infrastructure required to conduct and analyze TRIM.FaTE simulations. The framework allow users to:

- Define the issue to be studied, including time period, geographic region, pollutants, media, and populations of interest;
- Specify and choose algorithms that will be used for simulations;
- Select modeling parameters, including emissions sources, characteristics of the environment (*e.g.*, air temperature and soil permeability), and simulation time step;
- Identify data sets to be used and created;
- Execute the simulation;
- Perform sensitivity studies; and
- Export results.

There are two versions of the framework: prototype and Version 1.0. The framework prototype has served as a testbed for evaluating approaches. It has been designed to allow changes to be quickly implemented and to allow ideas to be quickly tried. The prototype was used to conduct the simulations described in this document. Version 1.0 was completed September, 1999 and will be used for future studies. Version 1.0 incorporates lessons learned from Prototypes I through V with the addition of features that increase the usefulness of the system, such as management of multiple modeling scenarios, portability between Windows and UNIX, and improved ease of use and robustness.

This description of the TRIM.FaTE computer framework generally covers both the prototype and Version 1.0 with indications where necessary that descriptions apply to only one of the implementations. Additional information about the architecture and design of TRIM.FaTE Version 1.0 can be found in Fine et al. (1998a,1998b).

F.1 SOFTWARE ARCHITECTURE

Bass et al. (1998) provide the following definition:

> The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

The prototypes and Version 1.0 have different architectures, so they are described separately.

F.1.1 ARCHITECTURE OF THE PROTOTYPES

The prototypes are implemented in an object-oriented manner, with almost all important quantities implemented as objects/classes. These include:

- parcels;
- volume elements;
- compartments;
- chemicals;
- links;
- algorithms;
- parameters (input parameters and calculated parameters);
- runs; and
- projects.

In the prototypes, a project is constructed in a hierarchical fashion: first a parcel is created, then volume elements can be added "to" the parcel, and then compartments can be added to the volume elements. Links can be created manually or can be automatically determined based on the spatial adjacency information of the project.

When a run is initiated, the needed transition matrices, source term vectors, and initial condition vector are constructed from the modeled system. This process utilizes the link topology and algorithms associated with each link, in addition to the source specified for particular compartments and the implied source terms calculated based on any boundary air concentrations specified. The transition matrices and associated source term and initial condition vectors are used in successive calls to the differential equation solver (*i.e.,* LSODE), after which the predicted chemical mass in each compartment is available.

An expression evaluator is also included within the design of the prototypes. This is used to evaluate almost all algorithms and other needed calculated quantities (*e.g.*, distribution coefficients in soil for organics, which are calculated from properties of the chemical and the soil compartment). The expressions themselves are stored as strings, using an object-oriented syntax consistent with the overall object model used. These expressions are "compiled" when a run is performed, with the objects needed to calculate each expression obtained for subsequent calculation. This allows flexible naming of variables and the creation of numerous intermediate terms that can help provide insight into the finer details of a particular run. Further, it significantly improves the quality of output reports that can be produced. For example, detailed reports can be generated that show the exact equations used to calculate a given quantity, as well as the values of the terms used in its calculation. The successful implementation of such a system in the prototype made it possible to seriously consider, and ultimately decide upon, implementing a similar capability in the TRIM.FaTE Version 1.0.

F.1.2 VERSION 1.0 ARCHITECTURE

As shown in Figure F-1, the TRIM computer system architecture is complex but flexible, allowing it to be applied in developing each of the different TRIM modules. The architecture components used to describe TRIM are classified as those that primarily provide (1) functionality (rectangles), and (2) those that primarily provide data (ovals). However, each of the components except for external data sources provide both functionality and data. The architectural components that are implemented to some degree in Version 1.0 are depicted with shadows. This figure is designed to represent the relationships within the TRIM computer framework, rather than the data flow within the system. Therefore, the word along an arrow forms a sentence where the verb on the arrow connects the two architecture components at the end of an arrow. For example, in the upper left hand corner of the figure, the TRIM Core “invokes” Analysis and Visualization Tools. Each of the TRIM components shown in Figure F-1 are described below.

F.1.2.1 TRIM Core

The TRIM Core primarily provides services required by multiple TRIM components or to integrate those components. The following functions are provided by the Core:

- A mapping tool that shows TRIM spatial objects, such as volume elements, and arbitrary supplemental information supplied by the user, such as soil types. The map display will also allow users to specify the X-Y extent of TRIM.FaTE volume elements.

- Coordination of TRIM graphical user interface components. This includes allowing the user to invoke TRIM modules, such as TRIM.FaTE, and maintaining lists of open windows.

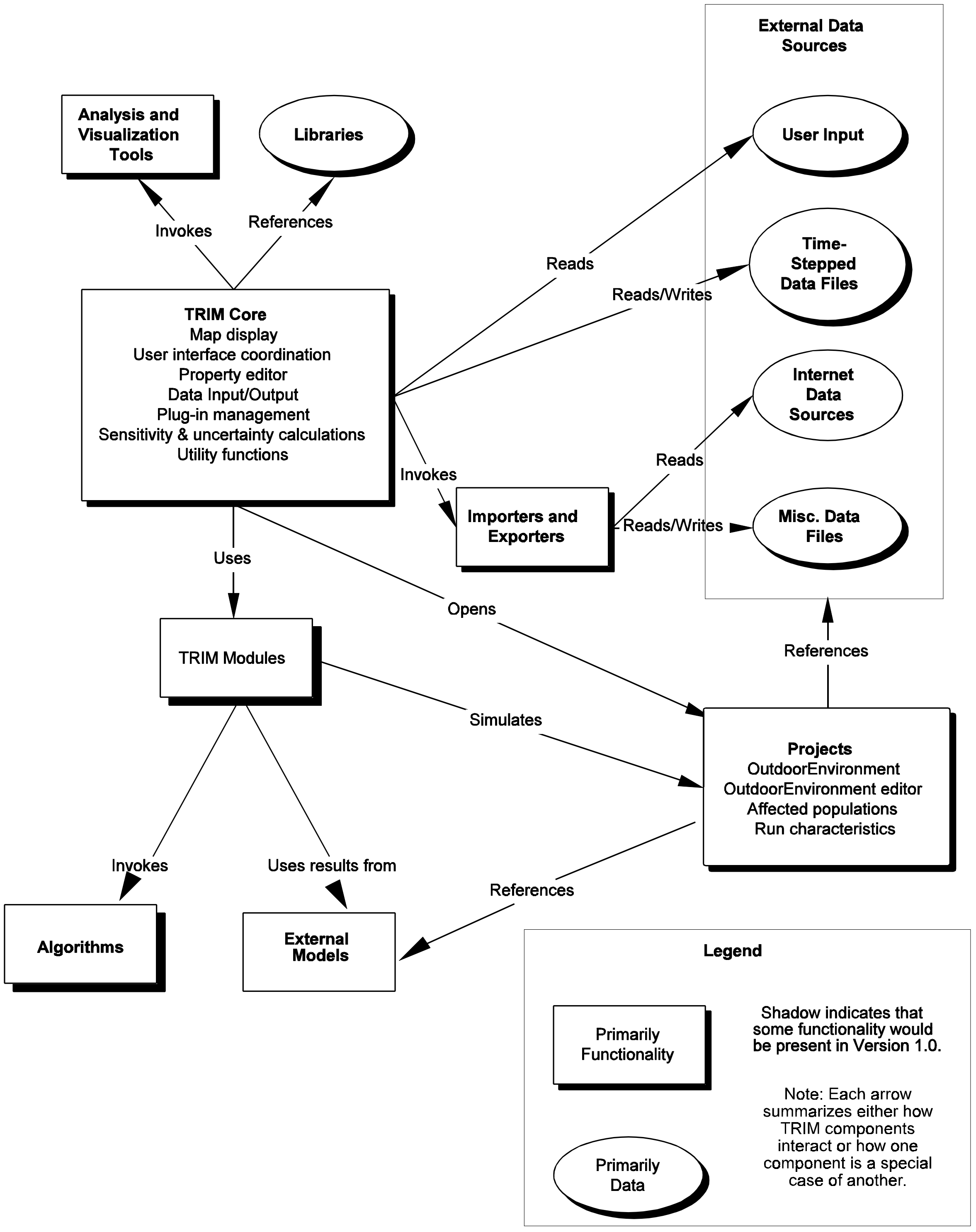
- Allowing users to edit and view property values, where a property is an attribute (*e.g.*, molecular weight) that describes an entity simulated by a model, such as a compartment or volume element. Properties include air temperature, scavenging coefficients, and chemical reaction rates.

- Management of plug-in data importers and exporters.

- Calculation of sensitivity and uncertainty using TRIM models (not supported in Version 1.0).

- Utility functions used by TRIM modules, such as routines to assist with data storage and retrieval.

Figure F-1
TRIM Computer System Architecture



F.1.2.2 Project

All information pertinent to an environmental study is stored in a “project.” Each project is also responsible for displaying the information it contains and allowing the user to change the information, in some cases relying on a TRIM Core functionality such as the property editor. A project can contain one or more “scenarios,” where each scenario contains a description of the outdoor environment being simulated, populations being studied, and model parameters, such as the simulation time step.

F.1.2.3 TRIM Modules

Each TRIM module, such as TRIM.FaTE, provides simulation or analysis functionality. Where required, they also provide specialized graphical user interfaces to support their functionality. Version 1.0 includes the TRIM.FaTE module. Future TRIM versions will have support for additional TRIM modules.

TRIM.FaTE uses a number of algorithms that compute chemical transfer coefficients between and transformation coefficients within conceptual compartments. As new chemicals and ecosystems are studied, new algorithms will be required. To address this need, users will be able to add algorithms, which are stored in libraries and projects. The algorithms that are stored in libraries can be applied to various projects.

F.1.2.4 Libraries

A substantial amount of relatively static information is required to conduct studies of multimedia fate and transport and effects on selected populations. For instance, the measured properties of chemicals change infrequently. Also, the boundaries of a study region might stay constant for years. Users can store such information in TRIM object libraries. They can then easily reuse selected information from a library in future projects. When information from a library is used in a project, a copy is made of the information, which protects the project from future changes to the library.

F.1.2.5 External Data Sources, Importers, and Exporters

It will be common for TRIM users to access or create data sets beyond TRIM projects. Some data sets may be too large to be conveniently stored in projects, while other data sets already exist in non-TRIM formats. TRIM provides several methods for accessing external information. The TRIM Core accepts user inputs and will read and write data in native TRIM files. The format of these files is based on the Environmental Decision Support System/Models-3 Input/Output Applications Programming Interface (I/O API) (Coats 1998). The I/O API format can be easily read and written from several programming languages, is platform-independent, is suitable for large data sets, is self-describing (*i.e.*, contains information about variables and time periods contained in the file), and is computationally efficient.

TRIM also allows users to plug in data importers and exporters. Data importers read non-TRIM data sets and set appropriate TRIM properties. For example, an importer could read files containing measurements of surface air temperature and set properties in ground-level TRIM air domains. Data exporters provide TRIM results in a form that is suitable for use by another program or for interactive review. This could include comma-delimited files that could be imported into a spreadsheet and tabular results for people to review.

F.1.2.6 Analysis and Visualization Tools

Version 1.0 includes no analysis and visualization tools. Instead, simulation results can be easily exported to Excel or other analysis packages. In the future, TRIM will include some analysis and visualization capabilities and might allow additional capabilities to be developed and plugged-in by users.

F.2 IMPLEMENTATION APPROACHES

The computer framework has been developed using an object-oriented approach. There has been much discussion in the software engineering literature (*e.g.,* Booch 1993) on the benefits of this approach, including increased software extensibility, reuse, and maintainability. The essence of object-oriented software development is that concepts, such as a volume element, are represented as a unit that contains internal data (*e.g.*, the boundaries of a volume element) and operations on that data (*e.g.*, compute volume) and that one class of objects (*e.g.*, volume element with vertical sides) can be a specialization of another class of objects (*e.g.*, volume element). Being able to specialize classes of objects allows general functionality to be shared by several specialized classes. TRIM's view of the outdoor environment (with volume elements that contain compartments) and the development of associated graphical user interfaces are well suited for an object-oriented treatment.

TRIM is being developed in an iterative manner. The major components and responsibilities of a class of objects are understood before implementation, but some details may be worked out as implementation proceeds. Graphical user interface mock-ups and significant new capabilities are shown to potential users before implementation begins. During implementation, the design is modified as needed. This user-oriented development approach helps highlight potential problems before undesirable approaches become embedded in the system. Furthermore, the object-oriented, open-ended structure of TRIM is intended to make future changes and additions a relatively simple process.

For Version 1.0 of TRIM, simpler and/or more reliable approaches were used in preference to faster and/or less resource-intensive approaches. In cases where simple approaches will not have adequate performance or will significantly limit the potential for future changes, more complex approaches will be used. As time permits, operations that cause noticeable speed or resource problems will be optimized.

F.3 IMPLEMENTATION LANGUAGE

Due to the different objectives of the framework prototypes and Version 1.0, different development languages were chosen. The rationale for each choice is described below.

F.3.1 PROTOTYPES

Microsoft's Visual Basic was used as the primary tool with which to implement the prototypes. This was due to a number of factors:

- Ease of use with Microsoft Excel, which all members on the team had (for early prototypes);
- Object-oriented features of language, while limited[1], simplify a dynamic, iterative architecture development cycle; and
- Straightforward to call needed Fortran codes (*e.g.*, differential equation solver, linear equation solver, triangulation).

F.3.2 VERSION 1.0

The Version 1.0 computer framework was developed primarily, but not entirely, in the Java programming language. Some parts of TRIM.FaTE, such as the differential equation solver, and other TRIM models, such as the exposure model, are implemented in FORTRAN, and other parts, such as the polygon overlay algorithm, are implemented in C.

Advantages of using Java include the following.

- Java code is portable across different hardware and operating systems. This is especially important for graphical user interfaces, which will comprise a large fraction of the TRIM code and which can be difficult to develop for multiple platforms.
- Java offers a good combination of speed of development, robustness, and support for object-oriented designs.
- Java is supported by multiple vendors. This often leads to competitive pressures to improve development tools, and it reduces the likelihood that one vendor's product strategy or financial problems will cripple TRIM development.
- Java provides built-in support for multithreading, which allows multiple operations to proceed simultaneously, and networking.

The disadvantages of using Java include the following:

- Java programs typically execute more slowly than programs written in C++ or BASIC. As the technologies for compiling and executing Java programs advance, the speed penalty for using Java should decrease.

[1] The primary limitation is that Visual Basic does not support inheritance. However, it does support polymorphism (an object/class can implement any number of interfaces), which is utilized to a large degree to simplify the logic of the programming.

- Fewer plug-in components (*e.g.*, mapping tools) and libraries (*e.g.*, matrix manipulation) are available for Java than there are for languages such as C++ or BASIC on Windows, but the number of Java components available is continuing to grow.

- Java development tools are not as mature as tools for other languages, but that situation is improving.

F.4 EMBEDDED TRANSFER ALGORITHMS

As described elsewhere, TRIM.FaTE allows users to specify and choose algorithms that compute chemical transformation and transfer factors. This provides significant flexibility to describe different pollutants and environmental systems. However, some transfer algorithms are too complex to be represented as user-entered formulas. These algorithms are described below.

F.4.1 WIND SPEED BETWEEN AIR COMPARTMENTS

The wind speed from one air compartment to another is calculated as the sum of the transport and dispersive/lateral wind speed. The methods used to calculate these quantities are implemented in subroutines within the source code, rather than through the use of expressions for the expression evaluator. Details on these methods can be found in Section 3.1 of the TRIM.FaTE TSD Volume II.

F.4.2 INTERFACIAL AREA BETWEEN VOLUME ELEMENTS

The interfacial area shared by volume elements is used frequently (*e.g.*, for advective and diffusive transfers). This is calculated by subroutines in the source code itself. In the prototype, each side of a volume element that might intersect another volume element is triangulated (in conjunction with a dynamic link library for triangulation). Next, intersection of the triangulations is computed. Version 1.0 uses a more specialized but faster approach that takes advantage of current restrictions on the structure of volume elements (sides must be vertical and tops and bottoms horizontal). The X-Y projections of side-by-side volume elements are examined for line segment overlap. The length of the overlap multiplies the extent of vertical overlap. The interfacial area for volume elements that are stacked vertically is computed by intersecting the polygons that represent the X-Y projections of the volume elements and then computing the area of the resulting polygon. When more general shapes are permitted for volume elements, a more general calculation, such as the triangulation approach, will be incorporated into Version 1.0.

F.5 REFERENCES

Bass, L., P. Clements, and R. Kazman. 1998. Software architecture in practice. Reading, MA: Addison-Wesley.

Booch, G. 1993. Object-oriented analysis and design with applications. Redwood City, California: The Benjamin/Cummings Publishing Company, Inc.

Coats, C. 1998. The EDSS/Models-3 I/O API. http://www.iceis.mcnc.org/EDSS/ioapi/.

Fine, S.S., A. Eyth, and H. Karimi. 1998a. The Total Risk Integrated Methodology (TRIM) Computer System Architecture. Research Triangle Park, NC: MCNC-North Carolina Supercomputing Center. November.

Fine, S.S., A. Eyth, and H. Karimi. 1998b. The Total Risk Integrated Methodology (TRIM) Computer System Design. Research Triangle Park, NC: MCNC-North Carolina Supercomputing Center. November.